What is claimed is:

1. A compiler apparatus that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the compiler apparatus comprising:

a parser unit operable to parse the source program;

an intermediate code conversion unit operable to convert the parsed source program into intermediate codes;

an optimization unit operable to optimize the intermediate codes so as to reduce a hamming distance between instructions placed in positions corresponding to the same instruction issue unit in consecutive instruction cycles, without changing dependency between the instructions corresponding to the intermediate codes; and

a code generation unit operable to convert the optimized intermediate codes into machine language instructions.

2. The compiler apparatus according to Claim 1,

wherein the optimization unit optimizes the intermediate codes by placing an instruction with higher priority in a position corresponding to each of the plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes, said instruction with higher priority having a smaller hamming distance from an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding cycle.

3. The compiler apparatus according to Claim 2,

wherein the optimization unit places an instruction with higher priority in the position corresponding to each of the plurality

of instruction issue units, said instruction with higher priority having a smaller hamming distance of an operation code from the instruction placed in the position corresponding to the same instruction issue unit in the immediately preceding cycle.

4. The compiler apparatus according to Claim 2,

wherein the optimization unit places an instruction with higher priority in the position corresponding to each of the plurality of instruction issue units, said instruction with higher priority having a smaller hamming distance of a register number from the instruction placed in the position corresponding to the same instruction issue unit in the immediately preceding cycle.

5. The compiler apparatus according to Claim 1,

wherein the optimization unit optimizes the intermediate codes by permuting instructions placed in a target instruction cycle within said cycle so as to reduce a hamming distance from an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding cycle, without changing dependency between the instructions corresponding to the intermediate codes.

6. The compiler apparatus according to Claim 5,

wherein the optimization unit permutes the instructions placed in the target instruction cycle within said cycle so as to reduce a sum of hamming distances from a plurality of instructions placed in positions corresponding to the plurality of instruction issue units in the immediately preceding cycle.

7. The compiler apparatus according to claim 5,

wherein the optimization unit permutes the instructions placed in the target instruction cycle within said cycle so as to

reduce a hamming distance of an operation code from the instruction placed in the position corresponding to the same instruction issue unit in the immediately preceding cycle.

8. The compiler apparatus according to Claim 5,

wherein the optimization unit permutes the instructions placed in the target instruction cycle within said cycle so as to reduce a hamming distance of a register number from the instruction placed in the position corresponding to the same instruction issue unit in the immediately preceding cycle.

9. A compiler apparatus that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the compiler apparatus comprising:

a parser unit operable to parse the source program;

an intermediate code conversion unit operable to convert the parsed source program into intermediate codes;

an optimization unit operable to optimize the intermediate codes so that a same register is accessed in consecutive instruction cycles, without changing dependency between instructions corresponding to the intermediate codes; and

a code generation unit operable to convert the optimized intermediate codes into machine language instructions.

10. The compiler apparatus according to Claim 9,

wherein the optimization unit optimizes the intermediate codes by placing an instruction with higher priority in a position corresponding to each of the plurality of instruction issue units, without changing dependency between the instructions

corresponding to the intermediate codes, said instruction with higher priority being for accessing a register of an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding instruction cycle.

11. The compiler apparatus according to Claim 10,

wherein the optimization unit places an instruction with higher priority in the position corresponding to each of the plurality of instruction issue units, said instruction with higher priority being for accessing a register with a register number close to a register number of the register of the instruction placed in the position corresponding to the same instruction issue unit in the immediately preceding cycle.

12. The compiler apparatus according to Claim 9,

wherein the optimization unit optimizes the intermediate codes by permuting instructions placed in a target instruction cycle within said cycle so that a register of an instruction placed in an immediately preceding cycle is accessed consecutively for a largest number of times, without changing dependency between the instructions corresponding to the intermediate codes.

13. A compiler apparatus that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units,

wherein an instruction which is to be issued with higher priority is predetermined for each of said plurality of instruction issue units, and

the compiler apparatus comprises:

a parser unit operable to parse the source program;

an intermediate code conversion unit operable to convert the parsed source program into intermediate codes;

an optimization unit operable to optimize the intermediate codes by placing said predetermined instruction with higher priority in a position corresponding to each of the plurality of instruction issue units, without changing dependency between instructions corresponding to the intermediate codes; and

a code generation unit operable to convert the optimized intermediate codes into machine language instructions.

14. The compiler apparatus according to Claim 13,

wherein the optimization unit optimizes the intermediate codes by permuting instructions placed in a target instruction cycle within said cycle so that the permuted instructions match a largest number of instructions in a set of instructions including the instruction which is issued with higher priority by each of the plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes.

15. A compiler apparatus that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the compiler apparatus comprising:

a parser unit operable to parse the source program;

an intermediate code conversion unit operable to convert the parsed source program into intermediate codes;

an optimization unit operable to optimize the intermediate codes by placing instructions in positions corresponding to the plurality of instruction issue units so that no instruction is placed in a position corresponding to a specific instruction issue unit out of

said plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes; and

a code generation unit operable to convert the optimized intermediate codes into machine language instructions.

16. A compiler apparatus that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the compiler apparatus comprising:

a parser unit operable to parse the source program;

an intermediate code conversion unit operable to convert the parsed source program into intermediate codes;

an interval detection unit operable to detect an interval in which no instruction is placed in a predetermined number of positions, out of a plurality of positions corresponding respectively to the plurality of instruction issue units in which instructions are to be placed, consecutively for a predetermined number of instruction cycles;

a first instruction insertion unit operable to insert, into immediately before the interval, an instruction to stop an operation of the instruction issue units corresponding to the positions where no instruction is placed; and

a code generation unit operable to convert the optimized intermediate codes into machine language instructions.

17. The compiler apparatus according to Claim 16, further comprising a second instruction insertion unit operable to insert, into immediately after the interval, an instruction to resume the operation of the instruction issue units corresponding to the

positions where no instruction is placed.

18. The compiler apparatus according to Claim 16,

wherein the processor includes a program status register that holds values indicating operation conditions of the plurality of instruction issue units, and

the instruction inserted by the first instruction insertion unit writes the values indicating the operation conditions of the plurality of instruction issue units into the program status register.

19. A compiler apparatus that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the compiler apparatus comprising:

a parser unit operable to parse the source program;

an intermediate code conversion unit operable to convert the parsed source program into intermediate codes;

an optimization unit operable to optimize the intermediate codes by placing instructions so as to operate only a specified number of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes; and

a code generation unit operable to convert the optimized intermediate codes into machine language instructions.

20. The compiler apparatus according to Claim 19,

wherein the source program includes unit number specification information specifying the number of instruction issue units used by the processor, and

the optimization unit optimizes the intermediate codes by

placing the instructions so as to operate only the instruction issue units of the number specified by the unit number specification information, without changing dependency between the instructions corresponding to the intermediate codes.

21. The compiler apparatus according to Claim 19, further comprising an acceptance unit operable to accept the number of instruction issue units used by the processor,

wherein the optimization unit optimizes the intermediate codes by placing the instructions so as to operate only the instruction issue units of the number accepted by the acceptance unit, without changing dependency between the instructions corresponding to the intermediate codes.

22. A compiler apparatus that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the compiler apparatus comprising:

a parser unit operable to parse the source program;

an intermediate code conversion unit operable to convert the parsed source program into intermediate codes;

an instruction scheduling unit operable to place instructions in positions corresponding to the plurality of instruction issue units without changing dependency between the instructions corresponding to the intermediate codes;

a register assignment unit operable to assign a register with higher priority to a variable, said register having a register number of a smaller hamming distance from a register number of a register for an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding instruction cycle,

and said variable being used by each of the instructions scheduled by the instruction scheduling unit; and

a code generation unit operable to convert the optimized intermediate codes into machine language instructions.

23. The compiler apparatus according to Claim 22,

wherein the register assignment unit assigns the register with higher priority to the variable used by each of the scheduled instructions, said register being same as the register for the instruction placed in the position corresponding to the same instruction issue unit in the immediately preceding cycle.

24. A compiler apparatus that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the compiler apparatus comprising:

a parser unit operable to parse the source program;

an intermediate code conversion unit operable to convert the parsed source program into intermediate codes;

an optimization unit operable to optimize the intermediate codes by placing instructions in positions corresponding to the plurality of instruction issue units according to each of a plurality of predetermined placement methods, without changing dependency between the instructions corresponding to the intermediate codes and by selecting one of the placed instructions which is expected to reduce power consumption most significantly during execution of a program from among the plurality of placed instructions; and

a code generation unit operable to convert the optimized intermediate codes into machine language instructions.

25. The compiler apparatus according to Claim 24,

wherein the optimization unit includes:

an instruction scheduling unit operable to place the instructions in the positions corresponding to the plurality of instruction issue units according to each of the plurality of predetermined placement methods, without changing dependency between the instructions corresponding to the intermediate codes; and

a register assignment unit operable to assign registers to variables according to each of a plurality of predetermined assignment methods, said variables being respectively used by the instructions scheduled by the instruction scheduling unit,

wherein the plurality of predetermined placement methods used by the instruction scheduling unit and the plurality of predetermined assignment methods used by the register assignment unit are searched using back tracking so as to obtain placement of the instructions which is expected to reduce power consumption most significantly during execution of the program.

26. The compiler apparatus according to Claim 25,

wherein the optimization unit further includes an instruction rescheduling unit operable to permute the instructions in the positions corresponding to the plurality of instruction issue units according to each of a plurality of predetermined permutation methods, said instructions using the variables to which the registers are assigned, and

the plurality of predetermined placement methods used by the instruction scheduling unit, the plurality of predetermined assignment methods used by the register assignment unit and the plurality of predetermined permutation methods used by the instruction rescheduling unit are searched using back tracking so as to obtain placement of the instructions which is expected to reduce

power consumption most significantly during execution of the program.

27. A compilation method for translating a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the compilation method comprising:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an optimization step of optimizing the intermediate codes so as to reduce a hamming distance between instructions placed in positions corresponding to the same instruction issue unit in consecutive instruction cycles, without changing dependency between the instructions corresponding to the intermediate codes; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

28. The compilation method according to Claim 27,

wherein in the optimization step, the intermediate codes are optimized by placing an instruction with higher priority in a position corresponding to each of the plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes, said instruction with higher priority having a smaller hamming distance from an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding cycle.

29. The compilation method according to Claim 27,

wherein in the optimization step, the intermediate codes are optimized by permuting instructions placed in a target instruction cycle within said cycle so as to reduce a hamming distance from an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding cycle, without changing dependency between the instructions corresponding to the intermediate codes.

30. A compilation method for translating a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the compilation method comprising:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an optimization step of optimizing the intermediate codes so that a same register is accessed in consecutive instruction cycles without changing dependency between instructions corresponding to the intermediate codes; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

31. The compilation method according to Claim 30,

wherein in the optimization step, the intermediate codes are optimized by placing an instruction with higher priority in a position corresponding to each of the plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes, said instruction with higher priority being for accessing a register of an instruction placed in a position corresponding to the same instruction issue unit in an

immediately preceding instruction cycle.

32. The compilation method according to Claim 30,

wherein in the optimization step, the intermediate codes are optimized by permuting instructions placed in a target instruction cycle within said cycle so that a register of an instruction placed in an immediately preceding cycle is accessed consecutively for a largest number of times, without changing dependency between the instructions corresponding to the intermediate codes.

33. A compilation method for translating a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units,

wherein an instruction which is to be issued with higher priority is predetermined for each of said plurality of instruction issue units, and

the compilation method comprises:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an optimization step of optimizing the intermediate codes by placing said predetermined instruction with higher priority in a position corresponding to each of the plurality of instruction issue units, without changing dependency between instructions corresponding to the intermediate codes; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

34. The compilation method according to Claim 33,

wherein in the optimization step, the intermediate codes are

optimized by permuting instructions placed in a target instruction cycle within said cycle so that the permuted instructions match a largest number of instructions in a set of instructions including the instruction which is issued with higher priority by each of the plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes.

35. A compilation method for translating a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the compilation method comprising:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an optimization step of optimizing the intermediate codes by placing instructions in positions corresponding to the plurality of instruction issue units so that no instruction is placed in a position corresponding to a specific instruction issue unit out of said plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

36. A compilation method for translating a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the compilation method comprising:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an interval detection step of detecting an interval in which no instruction is placed in a predetermined number of positions, out of a plurality of positions corresponding respectively to the plurality of instruction issue units in which instructions are to be placed, consecutively for a predetermined number of instruction cycles;

a first instruction insertion step of inserting, into immediately before the interval, an instruction to stop an operation of the instruction issue units corresponding to the positions where no instruction is placed; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

37. A compilation method for translating a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units,

wherein the source program includes unit number specification information specifying the number of instruction issue units used by the processor, and

the compilation method comprises:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an optimization step of optimizing the intermediate codes by placing the instructions so as to operate only the specified number of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

38. The compilation method according to Claim 37,

wherein in the optimization step, the intermediate codes are optimized by placing the instructions so as to operate only the instruction issue units of the number specified by the unit number specification information, without changing dependency between the instructions corresponding to the intermediate codes.

39. The compilation method according to Claim 37, further comprising an acceptance step of accepting the number of instruction issue units used by the processor,

wherein in the optimization step, the intermediate codes are optimized by placing the instructions so as to operate only the instruction issue units of the number accepted by the acceptance unit, without changing dependency between the instructions corresponding to the intermediate codes.

40. A compilation method for translating a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the compilation method comprising:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an instruction scheduling step of placing instructions in positions corresponding to the plurality of instruction issue units without changing dependency between the instructions corresponding to the intermediate codes;

a register assignment step of assigning a register with higher priority to a variable, said register having a register number of a

smaller hamming distance from a register number of a register for an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding instruction cycle, and said variable being used by each of the instructions scheduled by the instruction scheduling unit; and

a code generation unit operable to convert the optimized intermediate codes into machine language instructions.

41. A compilation method for translating a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the compilation method comprising:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an optimization step of optimizing the intermediate codes by placing instructions in positions corresponding to the plurality of instruction issue units according to each of a plurality of predetermined placement methods, without changing dependency between the instructions corresponding to the intermediate codes and by selecting one of the placed instructions which is expected to reduce power consumption most significantly during execution of a program from among the plurality of placed instructions; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

42. A program for a compiler that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions

executed respectively by the plurality of execution units, the program comprising:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an optimization step of optimizing the intermediate codes so as to reduce a hamming distance between instructions placed in positions corresponding to the same instruction issue unit in consecutive instruction cycles, without changing dependency between the instructions corresponding to the intermediate codes; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

43. The program according to Claim 42,

wherein in the optimization step, the intermediate codes are optimized by placing an instruction with higher priority in a position corresponding to each of the plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes, said instruction with higher priority having a smaller hamming distance from an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding cycle.

44. The program according to Claim 42,

wherein in the optimization step, the intermediate codes are optimized by permuting instructions placed in a target instruction cycle within said cycle so as to reduce a hamming distance from an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding cycle, without changing dependency between the instructions corresponding to the intermediate codes.

45. A program for a compiler that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the program comprising:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an optimization step of optimizing the intermediate codes so that a same register is accessed in consecutive instruction cycles, without changing dependency between instructions corresponding to the intermediate codes; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

46. The program according to Claim 45,

wherein in the optimization step, the intermediate codes are optimized by placing an instruction with higher priority in a position corresponding to each of the plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes, said instruction with higher priority being for accessing a register of an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding instruction cycle.

47. The program according to Claim 45,

wherein in the optimization step, the intermediate codes are optimized by permuting instructions placed in a target instruction cycle within said cycle so that a register of an instruction placed in an immediately preceding cycle is accessed consecutively for a

largest number of times, without changing dependency between the instructions corresponding to the intermediate codes.

48. A program for a compiler that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units,

wherein an instruction which is to be issued with higher priority is predetermined for each of said plurality of instruction issue units, and

the program comprises:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an optimization step of optimizing the intermediate codes by placing said predetermined instruction with higher priority in a position corresponding to each of the plurality of instruction issue units, without changing dependency between instructions corresponding to the intermediate codes; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

49. The program according to Claim 48,

wherein in the optimization step, the intermediate codes are optimized by permuting instructions placed in a target instruction cycle within said cycle so that the permuted instructions match a largest number of instructions in a set of instructions including the instruction which is issued with higher priority by each of the plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes.

50. A program for a compiler that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the program comprising:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an optimization step of optimizing the intermediate codes by placing instructions in positions corresponding to the plurality of instruction issue units so that no instruction is placed in a position corresponding to a specific instruction issue unit out of said plurality of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

51. A program for a compiler that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the program comprising:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an interval detection step of detecting an interval in which no instruction is placed in a predetermined number of positions, out of a plurality of positions corresponding respectively to the plurality of instruction issue units in which instructions are to be placed, consecutively for a predetermined number of instruction cycles;

a first instruction insertion step of inserting, into immediately before the interval, an instruction to stop an operation of the instruction issue units corresponding to the positions where no instruction is placed; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

52. A program for a compiler that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units,

wherein the source program includes unit number specification information specifying the number of instruction issue units used by the processor, and

the program comprises:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an optimization step of optimizing the intermediate codes by placing the instructions so as to operate only the specified number of instruction issue units, without changing dependency between the instructions corresponding to the intermediate codes; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

53. The program according to Claim 52,

wherein in the optimization step, the intermediate codes are optimized by placing the instructions so as to operate only the instruction issue units of the number specified by the unit number specification information, without changing dependency between the instructions corresponding to the intermediate codes.

54. The program according to Claim 52, further comprising an acceptance step of accepting the number of instruction issue units used by the processor,

wherein in the optimization step, the intermediate codes are optimized by placing the instructions so as to operate only the instruction issue units of the number accepted by the acceptance unit, without changing dependency between the instructions corresponding to the intermediate codes.

55. A program for a compiler that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the program comprising:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an instruction scheduling step of placing instructions in positions corresponding to the plurality of instruction issue units without changing dependency between the instructions corresponding to the intermediate codes;

a register assignment step of assigning a register with higher priority to a variable, said register having a register number of a smaller hamming distance from a register number of a register for an instruction placed in a position corresponding to the same instruction issue unit in an immediately preceding instruction cycle, and said variable being used by each of the instructions scheduled by the instruction scheduling unit; and

a code generation unit operable to convert the optimized intermediate codes into machine language instructions.

56. A program for a compiler that translates a source program into a machine language program for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units, the program comprising:

a parser step of parsing the source program;

an intermediate code conversion step of converting the parsed source program into intermediate codes;

an optimization step of optimizing the intermediate codes by placing instructions in positions corresponding to the plurality of instruction issue units according to each of a plurality of predetermined placement methods, without changing dependency between the instructions corresponding to the intermediate codes and by selecting one of the placed instructions which is expected to reduce power consumption most significantly during execution of a program from among the plurality of placed instructions; and

a code generation step of converting the optimized intermediate codes into machine language instructions.

57. A computer-readable recording medium for recording machine language codes for a processor including a plurality of execution units which can execute instructions in parallel and a plurality of instruction issue units which issue the instructions executed respectively by the plurality of execution units,

wherein the machine language codes include a first instruction to stop an operation of any of the instruction issue units during an interval in which the instructions are not executed for a predetermined number of instruction cycles or more in said any of the instruction issue units.

58. The computer-readable recording medium according to Claim 57,

wherein the machine language codes further include a second instruction to resume the operation of said any of the instruction issue units just after the interval has passed.

59. The computer-readable recording medium according to Claim 57,

wherein the processor includes a program status register that holds values indicating operation conditions of the plurality of instruction issue units, and

the first instruction writes the values indicating the operation conditions of the plurality of instruction issue units into the program status register.